

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION

FOR

**TEXTURE MAPPING METHOD AND APPARATUS FOR COMPUTER IMPLEMENTED
GRAPHICAL IMAGE PROCESSING**

INVENTORS:

**DANIEL S. RICE
YAJYUN WANG**

PREPARED BY:

**LAW OFFICES OF JAMES D. IVEY
3025 TOTTERDELL STREET
OAKLAND, CALIFORNIA 94611-1742
(510) 336-1100**

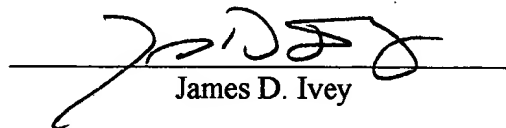
FILE NUMBER: P-2004.01

Certificate of Mail by Express Mail under 37 CFR § 1.10

EXPRESS MAIL LABEL NO.: EE 668 097 088 US

Date of Deposit: May 10, 1999

I hereby certify that this paper or fee is being deposited with the U.S. Postal Service
"Express Mail Post Office to Addressee" service under 37 CFR § 1.10 on the date
indicated above and is addressed to Box: PATENT APPLICATION, Assistant
Commissioner for Patents, Washington, D.C. 20231.


James D. Ivey

SPECIFICATION

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223

of the picture, i.e., of the rendering in a computer display device, of a graphical object and has a single color. A texel is an element of a texture image and similarly has a single color. Mapping a texture image to a graphical object which is either not flat or not two-dimensional almost always results in misalignment of texels of the mapped texture image and pixels of the rendered image of the graphical object, i.e., results in texels and pixels which are not coincident in a common coordinate space. In addition, it is frequently desirable to scale a textured graphical object to a particular size and to scale the texture image accordingly. The scaling of a textured image can be used, for example, to represent distance from a viewer of the textured graphical object in a computer display device. Such scaling requires magnification or minification of the textured image to comport with the relative size of the graphical object, resulting in further misalignment of pixels and texels.

To combine the color of a pixel with the color of a texel in any of the ways listed above, coordinates of the pixel in the coordinate space of the texture image are determined and the color of the texel at those coordinates is retrieved from the texture image so that the colors can be combined. When a pixel is not aligned with any particular texel, the color of a texel corresponding to the pixel is generally derived from the texels nearest the pixel. In some conventional texture mappers, the color of the texel nearest the pixel is selected. However, mapping a texel to a pixel in this manner frequently results in undesirable effects in the rendering of the textured graphical object. It is usually preferred to interpolate a color between the colors of the texels nearest the pixel. Mapping an interpolated texel to the pixel and combining the color of the interpolated texel with the color of the pixel generally achieves a smooth, visually pleasing, desirable result.

In conventional texture mapping, the proximity of a pixel of the graphical object to each of the nearest four texels of the texture image is determined, and the weighted average of the colors of the four nearest texels is determined, based on the relative proximity of each of the texels to the pixels. The relative proximity of each of the texels to a particular pixel is calculated by (i) determining the distance between each of the texels and the pixel, (ii) scaling the determined distances to produce respective weights for the texels, (iii) multiplying each component of the

color of each texel by the respective weight of the texel, and (iv) summing the weighted components of the colors to form a weighted average color. Scaling each of the determined distances typically involves several arithmetic operations, requiring substantial computing resources.

5 For example, calculation of complementary weights for two texels near a pixel require two distance calculations (each typically involving at least a subtraction operation), and a separate multiplication operation for each distance calculation. These operations are required for deriving a weighted average of only two texels in a single dimension. Typically, a weighted average of four texels in two dimensions is combined with the pixel. The most efficient of conventional texture mapping mechanisms repeats the above operations to form a weighted average of the remaining two texels and then repeats the above operations again to form an interpolated texel color which is a weighted average of the two previously calculated weighted averages. These operations represent a substantial component of the resources required to map a texture image to a graphical object.

10 Because of the substantial resources required in texture mapping a graphical image, a need for ever increasingly efficient texture mapping systems persists in the industry.

SUMMARY OF THE INVENTION

15 In accordance with the present invention, pixels of a graphical object are mapped to a coordinate space of a texture image and a weighted average color of the four nearest texels of the texture image is blended with the color of each pixel to give a rendering of the graphical object a textured appearance of the texture image. To calculate the weighted average of the colors of two texels near a particular pixel, a fractional portion of a texture coordinate, e.g., the horizontal texture coordinate, of the pixel is determined and a pair of complementary coefficients is retrieved from a table of pairs of predetermined, complementary coefficients according to the fractional portion of the coordinate of the pixel. Each of the complementary coefficients corresponds to the relative distance between the pixel and each of the two texels as represented by the fractional

portion of the first coordinate of the pixel in the coordinate space of the texture image since each texel has whole, integer coordinates in the coordinate space of the texture image. Each coefficient of the pair of complementary coefficients is used to weight a respective one of the colors of the two texels and the weighted colors are summed to produce a weighted average color of the two texels. A weighted average of the colors of the other two of the four nearest texels is calculated in the same manner.

Since a pair of complementary coefficients are retrieved from a table according to the fractional portion of the texel coordinate of the pixel, distance calculations between texels and the pixel, and multiplication operations to weight the distances, are obviated. Instead, a single value, i.e., the fractional portion of the texel coordinate of the pixel, is calculated and two (2) appropriately weighted, complementary coefficients are retrieved in a single read operation. Therefore, the resources required to determine the relative weights of the nearest texels is substantially reduced.

To calculate the weighted average of the colors of all four nearest texels, a fractional portion of a second coordinate, e.g., the vertical coordinate, of the pixel is determined and a second pair of complementary coefficients is retrieved from the same table according to the fractional portion of the second coordinate. Each of the pair of complementary coefficients is used to weight a respective one of the weighted average colors and the two weighted average colors are then summed to produce an interpolated texel color which is a weighted average of the colors of the four texels nearest the pixel.

The pair of complementary coefficients are partitioned values in a single data word and are therefore loaded into the processor which performs the calculate the weighted average only once to weight two separate colors. As a result, for each pixel rendered, one load instruction to load the second of two separate coefficients is obviated.

In one embodiment of the present invention, the precision and data format of each coefficient of the table are the same as the precision and data format of each component of each pixel of the rendered graphical image. Accordingly, the processing environment remains unchanged while a computer processor alternately interpolates a texel color and combines the

interpolated texel color with the color of the pixel. For example, in a processor in which a scale factor defines the precision and data format of operands of operations performed in the interpolation of the color of a texel and the combination of the interpolated texel color with the color of the pixel, a single scale factor defines the precision and data format of both the coefficients stored in the table and the components of the color of the pixel. As a result, changing of the operational environment, which typically involves at least a store instruction, during interpolation of colors of texels and combination of the interpolated colors of texels with the color of the pixel is obviated, thereby further increasing the efficiency with which a texture graphical image is rendered.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system which includes a processor and a texture mapper in accordance with the present invention.

Figure 2 is a block diagram showing the processor of Figure 1 in greater detail.

Figure 3 shows a pixel in a texel coordinate space.

Figure 4 is a block diagram of the texture mapper of Figure 1 in greater detail.

Figure 5 is a block diagram of a record of a texture coordinate of a pixel.

Figures 6 and 7 are each a block diagram illustrating a partitioned multiplication operation performed by the processor of Figure 2.

Figure 8 is a block diagram illustrating a partitioned addition operation performed by the processor of Figure 2.

Figures 9A and 9B are block diagrams illustrating a partitioned packing operation performed by the processor of Figure 2.

Figure 10 is a diagrammatic view of multiple texture sub-images of a mipmap texture image.

Figure 11 illustrates the location of a pixel between levels in a mipmapping texture coordinate space.

Figure 12 is a block diagram of a weighted level record of the pixel of Figure 11.

DETAILED DESCRIPTION

5 In accordance with the present invention, relative weights of texels nearest a particular pixel are determined by a table lookup of predetermined weights according to an integer which represents a fractional portion of a texel address. A partitioned coefficient, which represents the relative weights of two texels, is retrieved from the weight table and used in a partitioned multiplication operation in a processor to weight each of four components of the color of a particular texel simultaneously and in parallel. The four components of the color can be, for example, alpha, blue, green, and red. By using a partitioned coefficient, the coefficient is loaded into the processor once for two consecutive partitioned multiplication operations obviating reloading the partitioned coefficient into the processor.

10 Furthermore, the partitioned coefficient is scaled such that a scale factor, which is loaded into the processor for a separate and independent calculation, can be left in the processor unchanged during calculations of the weighted average of the nearest texels. As a result, significant processing involved in swapping different scale factors into and out of the processor is avoided.

Hardware Components of the Texture Mapping System

20 To facilitate appreciation of the present invention, the hardware components of the texture mapping system are briefly described. Computer system 100 (Figure 1) includes a processor 102 and memory 104 which is coupled to processor 102 through a bus 106. Processor 102 fetches from memory 104 computer instructions and executes the fetched computer instructions. Processor 102 also reads data from and writes data to memory 104 and sends data and control signals through bus 106 to a computer display device 108 in accordance with fetched and executed computer instructions. Processor 102 is described in greater detail below.

25 Memory 104 can include any type of computer memory and can include, without limitation, randomly accessible memory (RAM), read-only memory (ROM), and storage devices which include storage media such as magnetic and/or optical disks. Memory 104 includes a texture mapper 110, which is a computer process executing within processor 102 from memory

104. A computer process is a collection of computer instructions and data which collectively define a task performed by computer system 100. As described more completely below, texture mapper 110 (i) reads from a graphical object 112 and a texture image 114, both of which are stored in memory 104, (ii) creates from graphical object 112 and texture image 114 a textured graphical object 116, and (iii) causes textured graphical object 116 to be displayed in computer display device 108.

Computer display device 108 can be any type of computer display device including without limitation a cathode ray tube (CRT), a light-emitting diode (LED) display, or a liquid crystal display (LCD). Computer display device 108 receives from processor 102 control signals and data and, in response to such control signals, displays the received data. Computer display device 108, and the control thereof by processor 102, is conventional.

Processor 102 is shown in greater detail in Figure 2 and is described briefly herein and more completely in United States patent application serial number 08/236,572 by Timothy J. Van Hook, Leslie Dean Kohn, and Robert Yung, filed April 29, 1994 and entitled "A Central Processing Unit with Integrated Graphics Functions" (the '572 application) which is incorporated in its entirety herein by reference. Processor 102 includes a prefetch and dispatch unit (PDU) 46, an instruction cache 40, an integer execution unit (IEU) 30, an integer register file 36, a floating point unit (FPU) 26, a floating point register file 38, and a graphics execution unit (GRU) 28, coupled to each other as shown. Additionally, processor 102 includes two memory management units (IMMU & DMMU) 44a-44b, and a load and store unit (LSU) 48, which in turn includes data cache 120, coupled to each other and the previously described elements as shown. Together, the components of processor 102 fetch, dispatch, execute, and save execution results of computer instructions, e.g., computer instructions of texture mapper 110 (Figure 1), in a pipelined manner.

PDU 46 (Figure 2) fetches instructions from memory 104 (Figure 1) and dispatches the instructions to IEU 30 (Figure 2), FPU 26, GRU 28, and LSU 48 accordingly. Prefetched instructions are stored in instruction cache 40. IEU 30, FPU 26, and GRU 28 perform integer, floating point, and graphics operations, respectively. In general, the integer operands and results are stored in integer register file 36, whereas the floating point and graphics operands and results

are stored in floating point register file 38. Additionally, IEU 30 also performs a number of graphics operations, and appends address space identifiers (ASI) to addresses of load/store instructions for LSU 48, identifying the address spaces being accessed. LSU 48 generates addresses for all load and store operations. The LSU 48 also supports a number of load and store operations, specifically designed for graphics data. Memory references are made in virtual addresses. MMUs 44a-44b map virtual addresses to physical addresses.

PDU 46, IEU 30, FPU 26, integer and floating point register files 36 and 38, MMUs 44a-44b, and LSU 48 can be coupled to one another in any of a number of configurations as described more completely in the '572 application. As described more completely in the '572 application with respect to Figures 8a-8d thereof, GRU 28 performs a number of distinct partitioned multiplication operations and partitioned addition operations. Various partitioned operations used by texture mapper 110 (Figure 1) are described more completely below.

As described above, processor 102 includes four (4) separate processing units, i.e., LSU 48, IEU 30, FPU 26, and GRU 28. Each of these processing units is described more completely in the '572 application. These processing units operate in parallel and can each execute a respective computer instruction while others of the processing units executes a different computer instruction. GRU 28 executes the partitioned multiplication and partitioned addition operations described below. As described in the '572 application, GRU 28 has two separate execution paths and can execute two instructions simultaneously. GRU 28 can execute a partitioned addition operation while simultaneously executing a partitioned multiplication operation.

In one embodiment, processor 102 is the UltraSPARC processor available from SPARC International, Inc., and computer system 100 (Figure 1) is the UltraSPARCstation available from Sun Microsystems, Inc. of Mountain View, California. Sun, Sun Microsystems, and the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

Texture Mapping

5 In creating textured graphical object 116 (Figure 1), texture mapper 110 (i) maps pixels of graphical object 112 to the coordinate space of texture image 114 to determine a texel color corresponding to each pixel and (ii), using one of a number of techniques, combines the texel
10 colors with the color of each of the pixels of graphical object 112 to form a corresponding pixel of textured graphical object 116. The result of the combination of colors from texels of texture image 114 with colors of corresponding pixels of graphical object 112 is textured graphical object 116 which can be, for example, of the general color and three-dimensional shading of graphical object 112 but has a textured appearance as defined by texture image 114. For example, if graphical object 112 is a green triangle and texture image 114 is a brick pattern, textured graphical object 116 appears to be made of green bricks.

Figure 3 shows a pixel p_1 of graphical object 112 (Figure 1) which is mapped by texture mapper 110 to a coordinate space 114CS (Figure 3) in which texture image 114 (Figure 1) is defined. As shown in Figure 3, pixel p_1 is nearest texels t_1 , t_2 , t_3 , and t_4 of texture image 114 (Figure 1). Accordingly, texture mapper 110 blends the color of pixel p_1 (Figure 3) with a texel color interpolated from texels t_1 , t_2 , t_3 , and t_4 . The interpolated texel color can be, e.g., a weighted average in which the weight attributed to each of texels t_1 , t_2 , t_3 , and t_4 corresponds to the distance between each of texels t_1 , t_2 , t_3 , and t_4 and pixel p_1 .

In conventional texture mappers, a weighted average color is calculated using floating point arithmetic operations at a significant cost in terms of processing resources and time. However, in accordance with the present invention, a weighted average color is determined using scaled weights which are represented by pairs of predetermined, complementary coefficients and which are retrieved from a weight table 402 (Figure 4), which is included in texture mapper 110.

25 In general, graphical object 112 (Figure 1) is specified by a number of points each of which is defined in a coordinate space and has a corresponding color. In general, the coordinate space of the points of graphical object 112 include coordinates x , y , z , u , and v , in which (x, y, z) specifies a location in three-dimensional space and (u, v) specify a location in the coordinate space of texture image 114. In rendering graphical object 112 to form textured graphical object 116 in

computer display device 108, texture mapper 110 interpolates individual pixels and corresponding colors from the points which collectively specify graphical object 112. In interpolating a particular pixel from a number of points of graphical image 116, texture mapper 110 interpolates texture coordinates of pixel p_1 in coordinate space 114CS of texture image 114, i.e., in the form of (u_p, v_p) . Texture mapper 110 stores the texture coordinates of pixel p_1 in records ipxu and ipxv, each of which stores a 32-bit signed number. Record ipxu stores the u coordinate of pixel p_1 , and record ipxv stores the v coordinate of pixel p_1 . Records ipxu and ipxv are directly analogous to one another and the following description of record ipxu is equally applicable to record ipxv.

Record ipxu is shown in greater detail in Figure 5. Record ipxu stores an integer value in which an implicit decimal point separates a whole, integer portion 506 and a fractional portion 508. Thus, processor 102 can use integer operations to process data which represent numbers with fractional components. The value stored in record ipxu is scaled such that (i) a value of $n-1$, where n is the number of texels in the dimension of the u coordinate of texture image 114, is stored in record ipxu if pixel p_1 has a u coordinate which is equal to the maximum value of u defined for texture image 114 in coordinate space 114CS and (ii) a value of 0 is stored in record ipxu if pixel p_1 has a u coordinate which is equal to the minimum value of u defined for texture image 114 in coordinate space 114CS. Record ipxu includes whole portion 506 and fractional portion 508, which are defined by records 502 (Figure 4) and 504 of texture mapper 110. Specifically, record 502 stores data having an integer value which represents the position within record ipxu of the least significant bit of whole portion 506, and record 504 stores data having an integer value which represents the position within record ipxu of the least significant bit of fractional portion 508. In one embodiment, fractional portion 508 includes four bits of record ipxu, and therefore represents fractions in units of sixteenths, and whole portion 506 includes at most 28 bits of record ipxu.

Texture mapper 110 parses whole portion 506 and fractional portion 508 using, for example, bitwise shifting and masking operations. Whole portion 506 specifies the u coordinate of texels t_1 and t_3 . The u coordinate of texels t_2 and t_4 is determined by incrementing the u

coordinate of texels t_1 and t_3 , respectively. Whole portion 506 can specify a u coordinate outside the range of u coordinates represented by texture image 114, e.g., outside the range 0 to $n-1$. A texel can be mapped to such a u coordinate in any of a number of ways. For example, (i) a specific, background color can be assigned to any texel whose u coordinate is less than 0 or greater than $n-1$, (ii) the color of the nearest texel can be selected such that the color of the texel whose u coordinate is 0 is used for all pixels whose u coordinate is less than 0 and the color of the texel whose u coordinate is $n-1$ is used for all pixels whose u coordinate is greater than or equal to n , or (iii) the color of the texel whose u coordinate is the u coordinate of pixel p_1 modulo n such that the texture pattern specified by texture image 114 is, in effect, repeated to cover the entire surface of graphical object 112.

Fractional portion 508 represents a fractional portion of the u coordinate of pixel p_1 and therefore specifies the relative weights of texels t_1 and t_2 and of texels t_3 and t_4 . Rather than calculating relative weights of texels t_1 - t_4 using floating point arithmetic operation, texture mapper 110 uses fractional portion 508 to retrieve a partitioned coefficient from a weight table 402. Each item of weight table 402 is partitioned into two 16-bit fixed point numbers representing the relative weight of a particular texel. In one embodiment, weight table 402 has sixteen items and fractional portion 508 has four bits which collectively specify one of the sixteen items of weight table 402. Table A below represents an illustrative example of the items of weight table 402.

Table A

| Fractional Portion 508 | Upper Weight (in hexadecimal) | Lower Weight (in hexadecimal) |
|------------------------|-------------------------------|-------------------------------|
| 0 | 4000 | 0000 |
| 1 | 3C00 | 0400 |
| 2 | 3800 | 0800 |
| 3 | 3400 | 0C00 |
| 4 | 3000 | 1000 |
| 5 | 2C00 | 1400 |

| | | |
|----|------|------|
| 6 | 2800 | 1800 |
| 7 | 2400 | 1C00 |
| 8 | 2000 | 2000 |
| 9 | 1C00 | 2400 |
| 10 | 1800 | 2800 |
| 11 | 1400 | 2C00 |
| 12 | 1000 | 3000 |
| 13 | 0C00 | 3400 |
| 14 | 0800 | 3800 |
| 15 | 0400 | 3C00 |

The partitioned coefficients of weight table 402 are selected such that the partitioned coefficients are substantially evenly distributed over the range of values of fractional portion 508 and such that the sum of the partitioned coefficients of each of the items of weight table 402 are substantially equal to the maximum weight of a color component. In one embodiment, a component of a color is represented by an eight-bit unsigned integer whose value can range from 0 to 255. In this embodiment, the sum of each pair of partitioned coefficients is therefore equal to 256.0.

Specific coefficients in Table A above are represented as hexadecimal numbers. Each of the coefficients specified in Table A have implicit decimal points which separate whole, integer portions and fractional portions. The particular number of bits in each of the whole and fractional portions is selected to match the particular data format of colors of pixels which are interpolated by texture mapper 110 from points of graphical object 112. As described more completely below, the particular format selected in one embodiment is one in which the ten (10) most significant bits define a whole, integer portion of a particular coefficient and the six (6) least significant bits define a fractional portion of the coefficient. In this embodiment which is represented in Table A,

the sum of the partitioned coefficients of each item is approximately 4000 in hexadecimal, which represents the value 256.0.

Texture mapper 110 (Figure 4) retrieves from weight table 402 an item 402F corresponding to fractional portion 508 and uses the partitioned coefficients of item 402F to
5 interpolate from the colors of texels t_1 , t_2 , t_3 , and t_4 (Figure 3) a composite weighted average color of a texel corresponding to pixel p_1 .

Processor 102 (Figure 1) can perform a number of partitioned operations, including MUL8X16, MUL8X16AL, MUL8X16AU, FPADD16, and FPACK16 operations. Texture mapper 110 multiplies each component of a color 602 (Figure 6) corresponding to texel t_1 by the
10 upper partitioned coefficient 402FU of item 402F by performance of the MUL8X16AU operation. Prior to performance of the MUL8X16AU operation, texture mapper 110 loads item 402F and color 602 into registers in floating point register file 38 (Figure 2) of processor 102. Color 602 has four components 602A, 602B, 602G, and 602R which correspond to alpha, blue, green, and red components, respectively, of a color. Each of components 602A, 602B, 602G, and 602R are eight-bit unsigned integers. Texture mapper 110 causes processor 102 to perform the MUL8X16AU operation to thereby multiply each of components 602A, 602B, 602G, and 602R by upper coefficient 402FU simultaneously and in parallel to produce a weighted color 604 having components 604A, 604B, 604G, and 604R. Weighted color 604 is a 64-bit word and each of components 604A, 604B, 604G, and 604R is a sixteen-bit fixed point number.

Texture mapper 110 multiplies each component of a color 702 (Figure 7) corresponding to texel t_2 by the lower coefficient 402FL of item 402F by causing processor 102 to perform the MUL8X16AL operation. Prior to performance of the MUL8X16AL operation, texture mapper 110 loads color 702 into a register in floating point register file 38 (Figure 2). Since item 402F is already loaded into a register in floating point register file 38 (Figure 2) for performance of the
25 MUL8X16AU operation as described above, texture mapper 110 does not re-load item 402F, thereby further improving the efficiency with which a texel is mapped to a particular pixel. Color 702 has four components 702A, 702B, 702G, and 702R which correspond to alpha, blue, green, and red components, respectively, of a color. Each of components 702A, 702B, 702G, and 702R

is an eight-bit unsigned integer. Texture mapper 110 uses the MUL8X16AL operation to cause processor 102 to multiply each of components 702A, 702B, 702G, and 702R by lower sixteen-bit fixed point number 402FL simultaneously and in parallel to produce a weighted color 704 having components 704A, 704B, 704G, and 704R. Weighted color 704 is a 64-bit word and each of components 704A, 704B, 704G, and 704R is a partitioned sixteen-bit fixed point number.

Texture mapper 110 uses the FPADD16 operation to add respective partitioned components of colors 604 and 704 to produce a weighted average color 802. In particular, performance of the FPADD16 operation by processor 102 (Figure 1) in accordance with computer instructions fetched from texture mapper 110 adds components 604A, 604B, 604G, and 604R to components 704A, 704B, 704G, and 704R, respectively, simultaneously and in parallel to produce respective components 802A, 802B, 802G, and 802R of weighted average color 802. Weighted average color 802 is a 64-bit word and each of components 802A, 802B, 802G, and 802R is a partitioned sixteen-bit fixed point number.

As described above, colors 602 (Figure 6) and 702 (Figure 7) include four components, each of which is a partitioned eight-bit unsigned integer, and weighted average color 802 includes four components, each of which is a partitioned sixteen-bit fixed point number. Texture mapper 110 (Figure 1) converts weighted average color 802 to the format of a color used by texture mapper 110, i.e., a 32-bit word which includes four partitioned eight-bit unsigned integers using the FPACK16 operation. The FPACK16 operation scales, clips, and packs each of four partitioned sixteen-bit fixed point numbers into a respective partitioned eight-bit unsigned integer and is represented diagrammatically in Figures 9A and 9B.

Floating point register file 38 (Figure 2) includes a graphics status register (GSR) 902 (Figure 9A) which in turn includes a scale factor 902S. As represented by logic block 904, each of components 802A, 802B, 802G, and 802R is bit-shifted to the left by the number of bits specified in scale factor 902S of GSR 902. For example, bit-shifting component 802A (Figure 9B) results in intermediate word 952. The seven least significant bits of intermediate word 952 represent a fractional portion of intermediate word 952 and the remaining most significant bits of intermediate word 952 represent the whole, integer portion of intermediate word 952. By

providing a particular value for scale factor 902S, the implicit decimal point of component 802A is, in effect, shifted to the right a distance specified by scale factor 902S from a default position. For example, the default position shown in Figure 9A immediately precedes the six least significant bits. In one embodiment, scale factor 902S stores data having a value of one, and component 802A therefore has an implicit decimal point immediately preceding the five least significant bits.

As represented by logic block 906, each of components 802A, 802B, 802G, and 802R, as bit-shifted, is clipped to produce a value between a maximum value, e.g., 255, and a minimum value, e.g., 0. For example, with respect to processing component 802A, a clip module 906A of logic block 906 compares the value of intermediate word 952 (Figure 9B) to a maximum value stored in a maximum record 956 and to a minimum value stored in a minimum record 958. Clip module 906A provides to a pack module 908A (i) the maximum value if the value of intermediate record 952 is greater than the maximum value stored in maximum record 956, (ii) the minimum value if the value of intermediate record 952 is less than the minimum value stored in minimum record 958, or (iii) the least significant eight bits of the whole, integer portion of intermediate record 952 if the value of the intermediate record 952 is between the maximum and minimum values.

As represented by logic block 908 (Figure 9A), each of components 802A, 802B, 802G, and 802R, as bit-shifted and clipped, is packed into a respective partitioned eight-bit component of weighted average color 910. Weighted average color 910 includes components 910A, 910B, 910G, and 910R, each of which is a partitioned eight-bit unsigned integer which is the preferred format of a color as used by texture mapper 110 as described above. Weighted average color 910 is a weighted average of colors 602 and 702 of texels t_1 and t_2 , respectively.

Texture mapper 110 repeats the process described above with respect to Figures 6, 7, 8, 9A, and 9B to produce a weighted average color from colors corresponding to texels t_3 and t_4 . Texels t_3 and t_4 are aligned vertically with texels t_1 and t_2 , respectively, whose relative weights are specified by respective partitioned coefficients of item 402F, which corresponds to the fractional portion 508 of the u coordinate of pixel p_1 . Accordingly, texture mapper 110 uses the partitioned

coefficients of item 402F to produce the weighted average color corresponding to texels t_3 and t_4 .

In a manner which is directly analogous to the weighted averaging of colors of texels t_1 and t_2 and of colors of texels t_3 and t_4 as described above, texture mapper 110 produces a interpolated texel color which is a weighted average of weighted average color 910 (Figure 9A) and the weighted average color corresponding to texels t_3 and t_4 . Specifically, texture mapper 110 (Figure 4) stores in record ipxv data which represent the v coordinate of pixel p_1 in coordinate space 114CS (Figure 3) of texture image 114. Texture mapper 110 (Figure 4) parses the data stored in record ipxv in the manner described above to produce a whole, integer portion and a fractional portion of record ipxv. The whole, integer portion of record ipxv specifies the v coordinate of texels t_1 and t_2 . Texels t_3 and t_4 have v coordinates which are one increment greater than the v coordinate of texels t_1 and t_2 . Of course, texture mapper 110 must generally determine texels t_1 , t_2 , t_3 , and t_4 , and therefore must generally determine the whole, integer portions of records ipxu and ipxv, prior to calculation of weighted average color 910 (Figure 9A) and the weighted average color corresponding to texels t_3 and t_4 .

Texture mapper 110 (Figure 4) retrieves from weight table 402 the item of weight table 402 corresponding to the fractional portion of record ipxv. Texture mapper 110 weights weighted average color 910 (Figure 9A), which is a weighted average of the colors of texels t_1 and t_2 using the upper sixteen-bit fixed number of the partitioned weight using the MUL8X16AU operation in the manner described above with respect to Figure 6. Similarly, texture mapper 110 (Figure 4) weights the second weighted average color, which is a weighted average of the colors of texels t_3 and t_4 using the lower sixteen-bit fixed number of the partitioned weight using the MUL8X16AL operation in the manner described above with respect to Figure 7. The results of the MUL8X16AU and MUL8X16AL operations are summed by texture mapper 110 using the FPADD16 operation in the manner described above with respect to Figure 8. The resulting partitioned sum is packed into four partitioned eight-bit unsigned integers using the FPACK16 operation as described above with respect to Figures 9A and 9B to produce an interpolated texel color.

Texture mapper 110 generates from the interpolated texel color and the color of pixel p_1 a

display pixel having a corresponding display color. As described briefly above, the interpolated texel color and the color of pixel p_1 by, for example, replace, modulation, blending, or decal techniques. In a preferred embodiment, texture mapper 110 combines the interpolated texel color and the color of pixel using the partitioned arithmetic operations described above to combine all four components of the colors simultaneously and in parallel within processor 102. Texture mapper 110 displays the generated display pixel color in computer display device 108 as a part of textured graphical object 116. In one embodiment, texture mapper 110 uses a Z buffer hidden surface removal mechanism and therefore stores the generated display pixel color and a corresponding z coordinate of pixel p_1 in a display buffer and Z buffer, respectively, if the contents of the Z buffer indicate that pixel p_1 is visible. Z buffer hidden surface removal mechanisms are well known and are not described further herein.

Texture mapper 110 performs the texture mapping technique described above for each pixel rendered of graphical object 112 (Figure 1) to produce and display in computer display device 108 textured graphical object 116. Texture mapper 110 renders graphical object 112 using conventional techniques to determine the location and color of each pixel of a graphical representation of graphical object 112. However, prior to causing display of each pixel of graphical object 112, texture mapper 110 maps each pixel to texture image 114 and blends colors of the four nearest texels in the manner described above to produce textured graphical object 116 and to give textured graphical object 116 the textured appearance defined by texture image 114.

Scaling of Partitioned Weights

Performance of the FPACK16 operation described above by processor 102 in response to a computer instruction of texture mapper 110 so directing scales, clips, and packs four partitioned sixteen-bit fixed point numbers into four partitioned eight-bit unsigned integer numbers simultaneously and in parallel. As further described above, performance of the FPACK16 operation scales by shifting each of the partitioned sixteen-bit fixed point numbers to the left by a number of bits specified in scale factor 902S (Figure 9A) of GSR 902 which is stored in floating point register file 38 (Figure 2). As described above, the FPACK16 operation is used both to

scale and pack weighted average colors as described above with respect to Figures 9A and 9B and to generate a display pixel color from the interpolated texel color and the pixel color. As described more completely above, scale factor 902S (Figure 9A) of GSR 902 specifies the location within a 16-bit word of an implicit decimal point which separates a whole, integer portion of the sixteen-bit word from a fractional portion of the sixteen-bit word.

In one embodiment, scale factor 902S is selected to specify that such a sixteen-bit word includes ten (10) most significant bits which specify a whole, integer portion and six (6) least significant bits which specify a fractional portion. In this embodiment, the most significant bit is an overflow bit, the next significant bit is a sign bit and the next eight bits specify, in conjunction with the sign bit, the integer portion of a number between -256.0 and 256.0. As a result, such a sixteen-bit word can represent all possible values of a color and yet maximizes precision in representing fractional components of a color.

The partitioned coefficients of the items of weight table 402 are stored within texture mapper 110 in the same format of operands used in combining interpolated texel colors with pixel colors. As a result, data which specifies that the implicit decimal point in a sixteen-bit word separates a ten-bit whole, integer portion and a six-bit fraction portion is loaded into GSR 902 only once during the rendering of numerous pixels of graphical object 112. Otherwise, the data stored in scale factor 902S would have to be changed prior to calculation of the interpolated texel color and again prior to generation of the corresponding display pixel color from the interpolated texel color and pixel p_1 . By avoiding storage of data in scale factor 902S twice for each pixel rendered, texture mapper 110 significantly improves the efficiency with which texture graphical object 116 is rendered from graphical object 112 and texture image 114.

Tri-linear Interpolation of Texel Color: Mipmapping

Texture mapping frequently requires that the texture image, e.g., texture image 114 (Figure 1), is scaled down or scaled up to minify or magnify, respectively, the texture pattern of the texture image in accordance with minification or magnification, respectively, of graphical object 112. For example, if graphical object 112 is defined to be at a position which is far from a

viewer, graphical object 112 is rendered as textured graphical object 116 with a relatively small size. To maintain relatively realistic texturing of textured graphical object 116, the texture pattern of texture image 114 must be minified.

Such minification is accomplished by a conventional technique called mipmapping. In mipmapping, texture image 114 (Figure 10) includes a number of texture sub-images 114A-G, each of which includes a graphical texture pattern which corresponds to a respective degree of minification of textured graphical object 116 (Figure 1). For example, texture sub-image 114A (Figure 10) includes a graphical texture pattern corresponding to graphical object 112 (Figure 1) in its original size, i.e., without minification or magnification. Texture sub-image 114B (Figure 10) has a width which is one-half the width of texture sub-image 114A, has a height which is one-half the height of texture sub-image 114A, and includes a graphical texture pattern which corresponds to graphical image 112 (Figure 1) when graphical image 112 is rendered to a size which is one-half the original size of graphical image 112. Each successive one of texture sub-images 114B-G (Figure 10) is one-half the width and one-half the height of the preceding one of texture sub-images 114A-G. By specifying a graphical texture pattern for several degrees of minification, undesirable artifacts of minifying the graphical texture pattern of texture image 114 are substantially and significantly reduced.

Graphical objects such as graphical object 112 (Figure 1) are frequently rendered at sizes with degrees of minification which are between the particular degrees of minification corresponding to texture sub-images 114A-G (Figure 10). As a result, in rendering a pixel of graphical object 112 (Figure 1), it is frequently desirable to interpolate, not only between the nearest texels in a particular texture sub-image, but also between the nearest two of texture sub-images 114A-G (Figure 10).

In accordance with the present invention, texture mapper 110 (Figure 1) uses a partitioned weight to interpolate between the two of texture sub-images 114A-G (Figure 10) nearest the pixel. Figure 11 shows pixel p_1 mapped between nearest texture sub-images 114B and 114C. When rendering graphical object 112 using a parallel view, the depth of pixel p_1 , i.e., the degree of minification of pixel p_1 , is less significant in the rendering of pixel p_1 than the u and v coordinates

of pixel p_1 in the coordinate space of a particular one of texture sub-images 114A-G. Therefore, in one embodiment, a degree of minification is calculated once during the rendering of graphical object 112 (Figure 1) with a parallel point of view and the calculated degree of minification is used in rendering all pixels of graphical object 112. Accordingly, partitioned coefficients of items of weight table 402 (Figure 4) are not used when rendering graphical object 112 with a parallel point of view. However, when graphical object 112 is rendered with a perspective point of view, the degree of minification is calculated more frequently and the partitioned coefficients of weight table 402 (Figure 4) are used to associate with adjacent texture sub-images of texture image 114 respective relative weights.

Texture mapper 110 calculates the degree of minification using well known relationships between the texture coordinates and device coordinates of computer display device 108 (Figure 1). The calculated degree of minification is stored within texture mapper 110 as a floating point number having a whole portion and a fractional portion. The value of the whole portion specifies the first of the nearest two of texture sub-images 114A-G (Figure 10). The other of the nearest two of texture sub-images 114A-G is the one corresponding to one increment of minification greater than that corresponding to the first of texture sub-images 114A-G. For example, the whole portion of the degree of minification of pixel p_1 (Figure 11) specifies texture sub-image 114B as the first of the two nearest texture sub-images, and texture sub-image 114C corresponds to a degree of minification one increment greater than that of texture sub-image 114B and is therefore the second of the two nearest texture sub-images.

The fractional portion of the calculated degree of minification of pixel p_1 is scaled from the range 0.0-1.0 to the range 0.0-256.0 in the format of a sixteen-bit fixed point number described above. The scaled fractional portion is stored in the lower partitioned coefficient 1202L (Figure 12) of a weighted level record 1202 within texture mapper 110 (Figure 4). Weighted level record 1202 (Figure 12) is a 32-bit word which includes two partitioned sixteen-bit fixed point coefficients. Texture mapper 110 (Figure 4) calculates a complementary scaled fractional portion, which is 256.0 minus the first scaled fractional portion, and stores the complementary scaled fractional portion in the upper partitioned coefficient 1202U of weighted level record 1202. The

sum of upper partitioned coefficient 1202U and lower partitioned coefficient 1202L is therefore 256.0 which is selected for the reasons given above with respect to the partitioned coefficient of items of weight table 402 (Figure 4).

5 In rendering pixel p_1 (Figure 11), texture mapper 110 (Figure 4) calculates a first composite weighted average color corresponding to the colors of texels t_1 (Figure 11), t_2 , t_3 , and t_4 based on the relative distances of texels t_1 , t_2 , t_3 , and t_4 from pixel p_{1B} , which is pixel p_1 projected into the plane of texture sub-image 114B. Texture mapper 110 (Figure 4) also calculates a second composite weighted average color corresponding to the colors of texels t_5 (Figure 11), t_6 , t_7 , and t_8 based on the relative distances of texels t_5 , t_6 , t_7 , and t_8 from pixel p_{1C} , which is pixel p_1 projected into the plane of texture sub-image 114C. Texture mapper 110 (Figure 4) calculates the first and second composite weighted average colors in the manner described above with respect to two-dimensional texture mapping. Texture mapper 110 produces a three-dimensional composite weighted average color, which is a weighted average of the first and second composite weighted averages, using weighted level record 1202 and the MUL8X16AU, MUL8X16AL, FPADD16, and FPACK16 operations in the manner described above with respect to Figures 6-9B. Since upper partitioned number 1202U (Figure 12) and lower partitioned number 1202L are scaled such that their sum is 256.0, scaling factor 902S (Figure 9A) of GSR 902 is not changed to produce, scale, clip, and pack the three-dimensional composite weighted average color. Texture mapper 110 (Figure 4) blends the three-dimensional composite weighted average color with the color of pixel p_1 (Figure 11) in the manner described above with respect to two-dimensional texture mapping to produce a pixel of textured graphical object 116 (Figure 1).

The above description is illustrative only and is not limiting. The present invention is limited only by the claims which follow.